A Content Management Layer for Software-Defined Information Centric Networks

Extended abstract for poster/demo session

Abhishek Chanda
WINLAB, Rutgers University
achanda@winlab.rutgers.edu

Cedric Westphal
Huawei Innovation Center
cedric.westphal@huawei.com

University of California Santa Cruz
cedric@soe.ucsc.edu

ABSTRACT

The traditional abstraction mechanism in Software Defined Networking does not support including non forwarding elements (proxy and cache for example) in a network. Thus, an OpenFlow network cannot support any content centric functionality natively. We propose an extension of the SDN abstractions to allow these functionalities. We describe our demonstration setup which works by extending OpenFlow to support content identification and management.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design,Network Protocols

Keywords

Information-Centric networking; Software Defined Networking; OpenFlow; Content Distribution.

1. INTRODUCTION

Software defined networking decouples the control and forwarding plane of a network. The most commonly used SDN technology is OpenFlow which defines abstract entities called *flow* as a set of parameters (source address, port, etc.).

While this abstraction works well in many scenarios, it faces serious problems in content centric networks [1] due to the following reasons: 1) OpenFlow does not provide abstractions for *content*. Per flow granularity does not work well for content: an operator might want to assign different forwarding rules to different content. 2) Content centric networks require non-forwarding elements like proxy and cache in a network. We would like these elements to be programmable in the same way switches and routers are. Currently, OpenFlow does not provide a way to include these elements in the network.

We have developed a content centric network architecture based on OpenFlow to solve these issues. Our architecture allows seamless integration of non-forwarding elements. This in turn allows the OpenFlow controller to take fine-grained forwarding decisions based on content. Our network has a number of cache elements which can store content and proxy elements to demultiplex connections from end-hosts. This work builds up on existing work like [6, 2]. Our goal is to demonstrate how to build an information-centric networking architecture atop an SDN infrastructure.

Our work also enables further extensions, such as support for content-based network coding within the managed domain [5], or support for having a *metadata-driven* mechanism to optimize parameters (like content size, or content access latency, etc.). Our framework would enable the network to extract such attributes at the network layer (without costly DPI) as in [3].

2. CACHING NETWORK ARCHITECTURE

Our proposed network can be transparently placed between a content provider network and the consumers. We separate out metadata from a content flow as it enters the network and put it in the control plane. In our implementation, the metadata consists of the file name parsed from the HTTP GET request and the TCP flow information. Thus, it is a five tuple of the form $\langle file\ name; dest\ IP; dest\ port; src\ IP; srcport \rangle$.

The system works as follows (Fig 1): 1) The controller installs static rules in the ingress switch to forward all HTTP traffic from the client to a proxy. The proxy terminates all TCP connections from the client for HTTP. 2) When the client issues an HTTP request, the proxy parses the request to extract the name of the content and the intended web server. 3) The proxy queries the controller with file name as a URI. 4) If the controller returns an IP and a port number, the proxy redirects the client's connection to that address. Otherwise, the proxy directs the connection to the original web server. 5) The controller determines whether the content from the web server should be cached. To cache the content, it computes a forking point where the connection from the web server to the proxy should be duplicated towards the cache. 6) The controller installs rules

ICN'13, August 12, 2013, Hong Kong, China.
ACM 978-1-4503-2179-2/13/08.

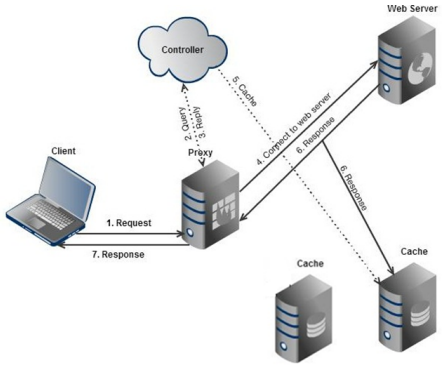

Figure 1: Demo setup: caches, proxy and client run on virtual machines and are connected to one software switch (OVS) and the web-server on another. Each instance of OVS runs on one physical box.

in the switches and invokes the cache. The controller notifies the cache of the content name and of the flow information to map the flow to the content. The controller also records the location of the cached content. 7) The cache saves the content with the file name obtained from the controller.

3. DEMO SETUP

We have implemented our proposed architecture. We demonstrate how the caching network routes content to a cache by modifying flow definitions at intermediate switches; the cache selection mechanism, whereby one can dynamically choose which cache to store content; and then we show future content access be retrieved by this cache, all controlled by an OpenFlow controller. The current testbed has a blade server (we would call it H) running Ubuntu [1]. The server runs an instance of Open vSwitch. H runs VMs for the cache, the proxy and the client. The FloodLight controller runs a module to do content based forwarding. We modified *tproxy* as the TCP proxy. The cache is written in C++, it listens on an Ethernet device on the VM and collects packets. When it sees a TCP FIN flag for a connection, it assembles data from the connection. It discards retransmitted packets, re-arranges packets based on sequence number and writes to a file on the disk. A python script is triggered by file write events on the disk. It gets the file name from the controller and saves the file in the disk.

To validate the set-up, we conducted some basic experiments. We placed a number of files in an external webserver and accessed them from our client VM. We measured the file download times for each file when it was served back from the actual webserver vs. when it was served back from the cache. See figure 2.

A content-aware architecture built on top of OpenFlow inherits its scalability issues. Those known issues are inherent to a centralized controller-driven system. [4] proposed DevoFlow where the controller only manages *significant flows*, thus reducing the control overhead. [7] proposed DIFANE which works by caching flows in switches. Since our abstraction allows us to treat content as a collection of flows, we can use the same flow optimization mechanisms. Primarily, we focus on caching flow definitions in the proxy and doing a distributed controller load balancing (for instance, requests for certain content can be sent to specific controllers).

[1]The demo can be run out of VMs on a laptop as well.

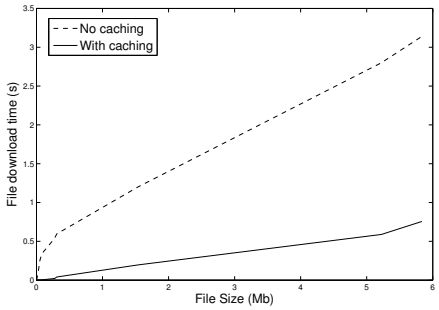

Figure 2: Variation of access delay with file size

We can envision some content driven mechanism to reduce load on the controller. We can classify content into three tiers as described below: 1) Most popular content for which the network will always have cached flow definitions at the switches and proxy so that incoming requests can be redirected to a known cache; 2) Moderately popular content for which the proxy will lookup a bloom filter first. If it returns a true, the proxy will query the controller. Otherwise the proxy will directly connect to the server; 3) Unpopular content which returns a false in the bloom filter.

In all cases, the controller can be periodically updated with content statistics, rather than for each request.

4. CONCLUSION

In this demo we show our proposed implementation of a content management layer on a software defined network. Effectively, we have proposed and evaluated a generalization of the SDN philosophy to include non-forwarding elements into the network with no modifications of client nor server. This approach enables the network operator to treat content in a differentiated manner, and to perform cache management and content routing using OpenFlow switches.

5. REFERENCES

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, July 2012.

[2] A. Chanda and C. Westphal. Content as a network primitive. In *arXiv:1212.3341 [cs.NI]*.

[3] A. Chanda, C. Westphal, and D. Raychaudhuri. Content based traffic engineering in software defined information centric networks. NOMEN'13. IEEE, 2013.

[4] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee. Devoflow: cost-effective flow management for high performance enterprise networks. Hotnets-IX, pages 1:1–1:6. ACM, 2010.

[5] M.-J. Montpetit, C. Westphal, and D. Trossen. Network coding meets Information-Centric networking. In *NOM Workshop, ACM MobiHoc'12*, June 2012.

[6] Y. Sakurauchi, R. McGeer, and H. Takada. Open web: Seamless proxy interconnection at the switching layer. In *Networking and Computing (ICNC), 2010*.

[7] M. Yu, J. Rexford, M. J. Freedman, and J. Wang. Scalable flow-based networking with difane. SIGCOMM '10. ACM, 2010.